

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 2002-091437

(43)Date of publication of application : 27.03.2002

(51)Int.Cl.

G10H 1/00
G06F 5/00
G10K 15/02
H03M 7/30
H04M 1/00

(21)Application number : 2000-283179

(71)Applicant : VICTOR CO OF JAPAN LTD

(22)Date of filing : 19.09.2000

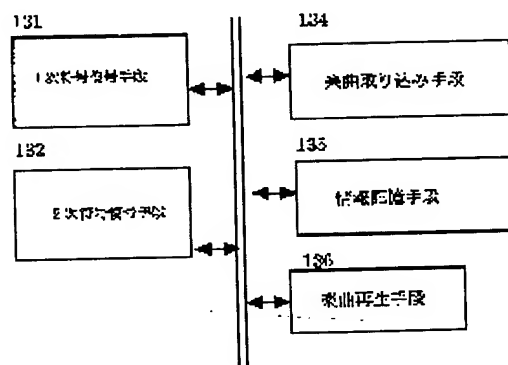
(72)Inventor : HIKAWA KAZUO

(54) PERFORMANCE INFORMATION COMPRESSING DEVICE AND PERFORMANCE INFORMATION DECODING DEVICE AND TELEPHONE TERMINAL EQUIPMENT

(57)Abstract:

PROBLEM TO BE SOLVED: To efficiently compress and store the data quantity of an incoming melody in a memory, and to start the reproduction of the incoming melody immediately after call incoming.

SOLUTION: The performance information of a piece of music to be reproduced at the time of call incoming is divided into not less than two blocks on a time base, and the first block is stored in a memory so as to be defrosted and performed immediately after the call incoming.



LEGAL STATUS

[Date of request for examination]

28.03.2003

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

Copyright (C): 1998,2003 Japan Patent Office

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2002-91437

(P2002-91437A)

(43) 公開日 平成14年3月27日 (2002.3.27)

(51)Int.Cl. ⁷	識別記号	F I	テ-リ-ト*(参考)
G 1 0 H 1/00	1 0 2	G 1 0 H 1/00	Z 5 D 3 7 8
			1 0 2 Z 5 J 0 6 4
G 0 6 F 5/00		G 0 6 F 5/00	H 5 K 0 2 7
G 1 0 K 15/02		G 1 0 K 15/02	
H 0 3 M 7/30		H 0 3 M 7/30	Z
審査請求 未請求 請求項の数10 O L (全 21 頁) 最終頁に続く			

(21) 出願番号 特願2000-283179(P2000-283179)

(22) 出願日 平成12年9月19日 (2000.9.19)

(71) 出願人 000004329

日本ビクター株式会社

神奈川県横浜市神奈川区守屋町3丁目12番地

(72) 発明者 飛河 和生

神奈川県横浜市神奈川区守屋町3丁目12番地 日本ビクター株式会社内

Fターム(参考) 5D378 MM28 MM62 MM65 MM68 MM96

QQ02 QQ06 QQ23 QQ24 QQ27

QQ38

5J064 AA02 BA15 BC01 BC02 BD02

BD03

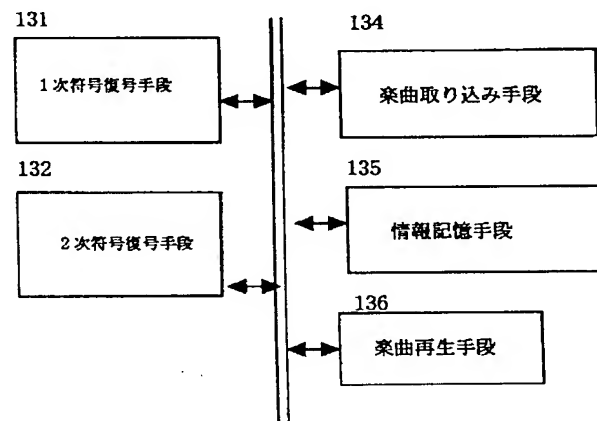
5K027 AA11 FF03 FF25

(54) 【発明の名称】 演奏情報圧縮装置、演奏情報復号装置及び電話端末装置

(57) 【要約】

【課題】 着信メロディのデータ量を効率的に圧縮してメモリに記憶すると共に、通話着信後直ちに着信メロディの再生を開始する。

【解決手段】 着信時に再生する楽曲の演奏情報を時間軸上で2つ以上のブロックに分割し、最初のブロックについては着信後すぐに解凍して演奏できる状態でメモリに記憶しておく。



【特許請求の範囲】

【請求項1】演奏情報を少なくとも音程の情報と、音の強さの情報と、音の長さの情報と、その他の情報とに分離し、前記各情報をそれぞれ独立した領域に配置した1次符号を生成する1次符号生成手段と、

前記1次符号生成手段により生成された1次符号の各領域の情報を圧縮する2次符号生成手段と、

前記演奏情報を時間軸上で2つ以上のブロックに分割し、少なくとも最初のブロックについては前記1次符号生成手段による圧縮及び／又は前記2次符号生成手段による圧縮を行わないことを特徴とする演奏情報圧縮装置。

【請求項2】前記1次符号生成手段は、各イベント間の相対時間の公約数、及び、各音符の長さの公約数を算出し、各イベント間の相対時間、及び、各音符の長さの値をそれぞれの公約数で除算した後符号化することを特徴とする請求項1に記載の演奏情報圧縮装置。

【請求項3】前記1次符号生成手段は、各音符の音程情報をそれ以前に出現した音符の音程の数値を使って一定の関数式に従って算出した数値と実際の音程の数値との残差で表すことを特徴とする請求項1又は2に記載の演奏情報圧縮装置。

【請求項4】前記1次符号生成手段は、各音符の強さ情報をそれ以前に出現した音符の強さの数値を使って一定の関数式に従って算出した数値と実際の強さの数値との残差で表すことを特徴とする請求項1乃至3のいずれか一項に記載の演奏情報圧縮装置。

【請求項5】前記1次符号生成手段は、特定の種類のイベントのパラメータ値をそれ以前に出現した同種類のイベントのパラメータ値を使って一定の関数式に従って算出した数値と実際のパラメータ値との残差で表すことを特徴とする請求項1乃至4のいずれか一項に記載の演奏情報圧縮装置。

【請求項6】前記1次符号生成手段は、前記各情報を当該領域においてトラック順に配置したことを特徴とする請求項1乃至5のいずれか一項に記載の演奏情報圧縮装置。

【請求項7】前記1次符号生成手段は、前記各領域をデータの性質が似ている領域同志が近くなるように配置したことを特徴とする請求項6に記載の演奏情報圧縮装置。

【請求項8】演奏情報を少なくとも音程の情報と、音の強さの情報と、音の長さの情報と、その他の情報とに分離し、前記各情報をそれぞれ独立した領域に配置した1次符号を復号する演奏情報復号装置であって、供給される前記1次符号を演奏情報に復号する1次符号復号手段を有することを特徴とする演奏情報復号装置。

【請求項9】演奏情報を記憶可能で、且つ前記演奏情報を電話着信時に再生する機能を有する電話端末装置において、

前記演奏情報を時間軸上で二つ以上のブロックに分割し、少なくとも最初のブロックが電話着信時に直ちに演奏できるように二番目以降のブロックのみを圧縮して記録することを特徴とする電話端末装置。

【請求項10】演奏情報を記憶可能で、且つ前記演奏情報を電話着信時に再生する機能を有する電話端末装置において、

前記演奏情報を時間軸上で二つ以上のブロックに分割して、それぞれのブロックを圧縮するときに、

少なくとも最初のブロックは、電話着信時に直ちに演奏でき、且つ、圧縮された二番目のブロックの復号を行うことができる時間の長さ及び／又は二番目以降のブロックの圧縮方法よりも短時間で復号可能な圧縮方法で圧縮されていることを特徴とする電話端末装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、演奏情報のデータ量を圧縮する演奏情報圧縮装置、復号を行う演奏情報復号装置及びこれらの装置を内蔵した電話端末装置に関する。

【0002】

【従来の技術】一般に、MIDI (Musical Instrument Digital Interface) データを保存する方式として、スタンダードMIDIファイル (以下、SMFという) が広く用いられ、このSMFでは図19に示すように個々の演奏情報がデルタ (Δ) タイムとイベントの2つの要素により構成されている。 Δ タイムは、隣合ったイベント間の相対時間を表し、イベントはノートオン又はノートオフのステータス、音程 (ノートナンバ) や音の強さ (ベロシティ) などの種々の演奏情報を含む。ここで、演奏情報とは音符により示される音程、音の長さの他に、音の強さ、楽曲の演奏上の拍子、テンポなどの情報、さらに音源の種類、リセットコントロールの情報などを含むものを指すものとする。また、このSMFでは各演奏情報が時間順に並んでトラックを構成している。ここで、SMF方式は記憶容量や伝送路の効率の利用という点ではあまり優れたものではないので、携帯電話の着信メロディや通信カラオケ、音楽データベースのように多数の楽曲データを記録、伝送するシステムでは、演奏情報のデータ量を効率的に圧縮することが求められている。

【0003】これと同時に、携帯電話の着信メロディでは、電話が着信したときに着信したことを着信メロディの再生によって知らせるが、この着信メロディのファイルをSMFファイルとして携帯電話本体に保存する際に、携帯電話本体の記憶容量の小さいメモリ部分に、できる限り多くの曲ファイルを記録することが求められている。

【0004】一方、テキスト等のデジタルデータを可逆的 (ロスレス) に圧縮する方法としては、文字列 (デー

3

タパターン) が繰り返すことを利用して繰り返し分を圧縮するLZ (Lempel-Zif) 法が現在広く使用され、LZ法は一般に使われている可逆式圧縮方法の中で、圧縮率が最も高いと言われている。LZ法は文字列が繰り返すことを利用して圧縮するので、入力データの中の限られた範囲内に出現する同一のデータパターンの数が多く、且つ同一データパターン長が長い場合に圧縮率が高くなるという性質を有する。

【0005】そこで、本出願人は、特開平9-16168において、LZ法により圧縮する前に予め、同一のデータパターンの長さが長く、出現回数が多く且つ近い距離で出現するように、演奏情報を音程と、強さと、長さその他の情報に分離し、それぞれ独立した領域に配置するように、演奏情報を少なくとも音程の情報と、音の強さの情報と、音の長さの情報とその他の情報に分離し、各情報をそれぞれ独立した領域に配置した1次符号を生成する1次符号生成手段と、前記1次符号生成手段により生成された1次符号の各領域の情報をLZ法により圧縮する2次符号生成手段とを有する演奏情報圧縮装置を提案した。

【0006】この装置では、1次符号生成手段が、各イベント間の相対時間の公約数及び各音符の長さの公約数を算出し、各イベント間の相対時間及び各音符の長さの値をこれらの公約数で除算した後符号化するよう構成されていることは前記提案の好ましい態様である。また、1次符号生成手段が、各音符の音程情報をそれ以前に出現した音符の音程の数値を使って一定の関数式に従って算出した数値と実際の音程の数値との残差で表すよう構成されていることは本発明の好ましい態様である。また、1次符号生成手段が、各音符の強さ情報をそれ以前に出現した音符の強さの数値を使って一定の関数式に従って算出した数値と実際の強さの数値との残差で表すよう構成されていることは前記提案の好ましい態様である。

【0007】また、1次符号生成手段が、特定の種類のイベントのパラメータ値をそれ以前に出現した同種類のイベントのパラメータ値を使って一定の関数式に従って算出した数値と実際のパラメータ値との残差で表すよう構成されていることは前記提案の好ましい態様である。また、1次符号生成手段が、各情報を当該領域においてトラック順に配置することは本発明の好ましい態様である。さらに、1次符号生成手段が、前記各領域をデータの性質が似ている領域同志が近くなるように配置することは前記提案の好ましい態様である。

【0008】

【発明が解決しようとする課題】最近の着信メロディはiモード等のサービスサイトよりダウンロードすることによって着信メロディファイルを得ることが主流になりつつあるが、ユーザは着信メロディ楽曲ファイル(以下ファイルと称する)の入手のために会費を支払ってお

4

り、携帯電話本体の内蔵メモリが少ないと、既にダウンロードしたファイルのファイルサイズの合計がメモリーの容量を超えた際に、以前に購入したファイルを消去しなければならない。また、この状況はユーザが自ら演奏データを入力する場合、あるいは専用のパーソナルコンピュータ等のソフトウェアを用いてファイルを携帯電話本体に記録する場合に於いても、同様な問題が発生する。また、今後は着信メロディのファイルフォーマットとしてSMF(スタンダードMIDIファイル)が用いられることが予想され、従来の専用ファイルよりも個々のファイルサイズが大きくなってしまふ点が問題点として指摘されている。

【0009】そこで、前記提案に記したようなファイル圧縮技術を用いることにより、ファイルサイズを小さくすることが容易に考えられるが、携帯電話に搭載されるCPUによって前記提案のような圧縮処理を行うと、再生しようとするファイルの全てを解凍し終わらない限り演奏の開始が行えないことと、実際の端末側の処理速度が遅いことにより、電話を着信してから着信メロディの再生音を出力することが可能となるまでに、かなりの時間がかかってしまうことが問題となっている。

【0010】本発明は上記従来の問題点に鑑み、そこで、予め携帯電話などの電話端末の内部・外部メモリーなどにあらかじめ指定された1つまたは複数のファイルを解凍した状態で保存することにより、電話着信時に即座に着信メロディを再生することを可能とする。

【0011】

【課題を解決するための手段】本発明は上記目的を達成するために、演奏情報を少なくとも音程の情報と、音の強さの情報と、音の長さの情報と、その他の情報に分離し、前記各情報をそれぞれ独立した領域に配置した1次符号を生成する1次符号生成手段と、前記1次符号生成手段により生成された1次符号の各領域の情報を圧縮する2次符号生成手段と、前記演奏情報を時間軸上で2つ以上のブロックに分割し、少なくとも最初のブロックについては前記1次符号生成手段による圧縮及び/又は前記2次符号生成手段による圧縮を行わないことを特徴とする演奏情報圧縮装置を提供する。

【0012】また、前記1次符号生成手段は、各イベント間の相対時間の公約数及び各音符の長さの公約数を算出し、各イベント間の相対時間及び各音符の長さの値をこれらの公約数で除算した後符号化することを特徴とする請求項1に記載の演奏情報圧縮装置を提供する。

【0013】更に、前記1次符号生成手段は、各音符の音程情報をそれ以前に出現した音符の音程の数値を使って一定の関数式に従って算出した数値と実際の音程の数値との残差で表すことを特徴とする請求項1又は2に記載の演奏情報圧縮装置を提供する。

【0014】更にまた、前記1次符号生成手段は、各音符の強さ情報をそれ以前に出現した音符の強さの数値を

使って一定の関数式に従って算出した数値と実際の強さの数値との残差で表すことを特徴とする請求項 1 乃至 3 のいずれか一項に記載の演奏情報圧縮装置を提供する。

【0015】また更に、前記 1 次符号生成手段は、特定の種類のイベントのパラメータ値をそれ以前に出現した同種類のイベントのパラメータ値を使って一定の関数式に従って算出した数値と実際のパラメータ値との残差で表すことを特徴とする請求項 1 乃至 4 のいずれか一項に記載の演奏情報圧縮装置を提供する。

【0016】また、前記 1 次符号生成手段は、前記各情報を当該領域においてトラック順に配置したことを特徴とする請求項 1 乃至 5 のいずれか一項に記載の演奏情報圧縮装置を提供する。

【0017】更に、前記 1 次符号生成手段は、前記各領域をデータの性質が似ている領域同志が近くなるように配置したことを特徴とする請求項 6 に記載の演奏情報圧縮装置を提供する。

【0018】更にまた、演奏情報を少なくとも音程の情報と、音の強さの情報と、音の長さの情報と、その他の情報に分離し、前記各情報をそれぞれ独立した領域に配置した 1 次符号を復号する演奏情報復号装置であって、供給される前記 1 次符号を演奏情報に復号する 1 次符号復号手段を有することを特徴とする演奏情報復号装置を提供する。

【0019】また更に、演奏情報を記憶可能で、且つ前記演奏情報を電話着信時に再生する機能を有する電話端末装置において、前記演奏情報を時間軸上で二つ以上のブロックに分割し、少なくとも最初のブロックが電話着信時に直ちに演奏できるように二番目以降のブロックのみを圧縮して記録することを特徴とする電話端末装置を提供する。

【0020】また、演奏情報を記憶可能で、且つ前記演奏情報を電話着信時に再生する機能を有する電話端末装置において、前記演奏情報を時間軸上で二つ以上のブロックに分割して、それぞれのブロックを圧縮するときに、少なくとも最初のブロックは、電話着信時に直ちに演奏でき、且つ、圧縮された二番目のブロックの復号を行うことができる時間の長さ及び又は二番目以降のブロックの圧縮方法よりも短時間で復号可能な圧縮方法で圧縮されていることを特徴とする電話端末装置を提供する。

【0021】

【発明の実施の形態】以下、図面を参照して本発明の実施の形態について説明する。図 1 は本発明に係る演奏情報圧縮装置の一例を示すブロック図、図 2 は図 1 の 1 次符号生成手段の一例を詳細に示すブロック図、図 3 は図 2 のチャンネル分離手段により作成されるチャンネルマップを示す説明図、図 4 は図 2 の解析手段の処理を説明するためのフローチャート、図 5 は図 2 の解析手段により作成されるノートテーブルを示す説明図、図 6 は図 2 の解

析手段により作成されるコントローラテーブルを示す説明図、図 7 は音符を表現する SMF の Δ タイムと本実施例のデュレーションの関係を示す説明図、図 8 は図 2 のノート Δ 符号生成手段の処理を説明するためのフローチャート、図 9 は図 2 のノート Δ 符号生成手段により生成されるノート Δ 符号を示す説明図、図 10 は図 2 のデュレーション符号生成手段の処理を説明するためのフローチャート、図 11 は図 2 のデュレーション符号生成手段により生成されるデュレーション符号を示す説明図、図 12 は図 2 のノートナンバ符号生成手段により生成されるノートナンバ符号を示す説明図、図 13 は図 2 のペロシティ符号生成手段により生成されるペロシティ符号を示す説明図、図 14 は図 2 のコントローラ符号生成手段により生成されるコントローラ符号を示す説明図、図 15 は SMF の連続イベントブロックを示す説明図、図 16 は本実施例の連続イベントブロックを示す説明図、図 17 は図 16 の連続イベントブロックの効果を示す説明図、図 18 は図 2 の符号配置手段により並べ替えられた 1 次符号を示す説明図、図 20 は、本発明による端末装置のうち、1 次符号化された状態で指定楽曲を保存する態様による端末の構成を示す説明図、図 21 は、本発明による端末装置のうち、1 次符号化も 2 次符号化もされない状態で指定楽曲を保存する態様による端末の構成を示す説明図、図 22 は、本発明による端末装置のうち、1 次符号化された状態で指定楽曲を追加的に保存する態様による端末の構成を示す説明図、図 23 は、本発明による端末装置のうち、1 次符号化も 2 次符号化もされない状態で指定楽曲を追加的に保存する態様による端末の構成を示す説明図である。

【0022】先ず、入力データ 1 は一例として図 7、図 19 に示すような SMF であり、SMF のフォーマットは Δ タイム、ステータス、ノートナンバ及びペロシティにより構成されている。ここで、本明細書では「発音開始イベント」を「ノートオンイベント」、「発音停止イベント」を「ノートオフイベント」と呼ぶ。また「ノートオンイベント」と「ノートオフイベント」を合わせて「ノートイベント」と呼び、それ以外の「イベント」を「コントローライベント」と呼ぶ。

【0023】図 1 において、SMF フォーマットの入力データ 1 は 1 次符号生成手段 2 により解析され、少なくとも演奏情報を音程と、強さと、長さその他の情報に分離され、各情報がそれぞれ独立した領域に配置した 1 次符号 3 が生成される。この 1 次符号 3 の各領域の符号は 2 次符号生成手段 4 により LZ 法で圧縮され、2 次符号 5 が生成される。なお、このように圧縮されたデータは図 20 ～ 図 28 に詳しく示す復号装置により LZ 法で復号され、音程と、強さと、長さその他の情報に基づいて音符が復元される。

【0024】1 次符号生成手段 2 は例えば図 2 に詳しく示すように、チャンネル分離手段 11 と、解析手段 12

7

と、ノートΔ符号生成手段13と、コントローラΔ符号生成手段14と、デュレーション符号生成手段15と、ノートナンバ符号生成手段16と、ベロシティ符号生成手段17と、コントローラ符号生成手段18と符号配置手段19で構成され、この例では1つの音符を示すノートΔ符号、コントローラΔ符号、デュレーション符号、ノートナンバ符号、ベロシティ符号及びコントローラ符号の6種類の1次圧縮符号が符号配置手段19により並べ換えられ、1次符号3として2次符号生成手段4に出力され、2次符号生成手段4により2次圧縮される。

【0025】チャンネル分離手段11ではSMF1の1トラックに複数チャンネルのイベントが含まれているか否かのチェックを行ない、複数チャンネルのイベントが含まれている場合には、1トラックの中に1チャンネルのイベントのみが含まれるように、トラックの分割を行う。そして、図3のようなトラックとチャンネル番号の対応を表したチャンネルマップを作成し、これ以降の処理はトラック単位で行う。ここで、SMFのイベントの大半はチャンネル情報を含んだものであるが、このようにトラック分割とチャンネルマップの作成を行うことにより、個々のイベントのチャンネル情報を省略することができ、データ量を削減することができる。

【0026】解析手段12では図4に示す処理を行い、トラック毎に図5に示すようなノートテーブルと図6に示すようなコントローラテーブルを作成する。まず、SMFから順次Δタイムとイベントを読み出し（ステップS1）、Δタイムからトラックの先頭を基準としたイベントの時間（以下では単に、イベントの時間と呼ぶ）を計算する（ステップS2）。次にイベントを解析し、イベントを「ノートオンイベント」、「ノートオフイベント」、「コントローライベント」の3種類に分類する。

【0027】そして、「ノートオンイベント」の場合には図5に示すようなノートテーブルにノートナンバ（音符の音程）とベロシティ（音符の強さ）を登録し（ステップS3→S4）、「ノートオフイベント」の場合にはデュレーション（音符の長さ）を計算してノートテーブルに登録する（ステップS5→S6）。また、「コントローライベント」の場合には図6に示すようなコントローラテーブルに登録し（ステップS7）、このようにして演奏情報毎にノートテーブルとコントローラテーブルを作成する（ステップS8→S1）。

【0028】ここで、ノートテーブルは図5に示すようにトラックのノート（音符）イベント情報を時間順に並べたものであり、コントローラテーブルは図6に示すように、トラックのコントローラ（音符以外）の情報を時間順に並べたものである。また、「ノートオンイベント」の場合にノートナンバとベロシティを書き込む際に、イベントの時間をノートテーブルの所定の欄に書き込み、また、ノートテーブルの「ノートオフ参照」欄を初期値として「0」にセットする。

8

【0029】また、イベントがノートオフであれば、ノートテーブルを先頭から走査して、ノートオフイベントの時間よりも早く、かつノートナンバが同じで、かつノートオフ参照欄が「0」にセットされているノートを選び出し、対応させる。そして対応するノートオンの時間Tonとノートオフの時間Toffとの差（Toff-Ton）を「デュレーション（音符の長さ）」とし、ノートテーブルの「デュレーション」欄に記録するとともに、「ノートオフ参照」欄を「1」にセットする。

10 【0030】ここで、「デュレーション」という概念はSMFにはないが、これを用いることによりノートオフイベントを省略できるので、データ容量が削減できる。SMFにおいて、1つの音符は図7のように1つのノートオンイベントと1つのノートオフイベントの組で表され、また、ノートオフイベントの前のΔタイムがデュレーションに相当する。ノートオフイベントのノートナンバは、ノートオンイベントとの対応を取る為に必要であり、デュレーションという概念を使ってノートオンとノートオフの対応を取っておけば不要である。

20 【0031】また、ノートオフイベントのベロシティは、MIDIデータを受け取るほとんどの音源が実際にはこの値を使用しないので、削除しても問題ない。したがって、ノートオフイベント（3バイト）を省略することにより、場合によってはΔタイムのデータ量が増えることもあるが、いずれにしてもノートオフイベント省略の効果の方が大きく、1つの音符につき最大3バイト分を削減することができる。その結果、1つの楽曲の中に1万個程度の音符が含まれている場合も珍しくないの
30 で、この場合には最大30Kバイトの削減ができることになり、圧縮効果が大きい。

【0032】イベントがノートオン、ノートオフ以外のイベントであれば、イベントの時間とイベントの内容をコントローラテーブルに登録し、このようにしてノートテーブルにはNA個のイベントが登録され、コントローラテーブルにはNB個のイベントが登録される。

40 【0033】次に、ノートΔ符号生成手段13とコントローラΔ符号生成手段14について説明する。この2つは、同じような処理内容であるので、以下ではノートΔ符号生成手段13を例に取って説明する。ノートΔ符号生成手段13は図8に示すように、まず、前述したノートテーブルに登録された各イベントに対し、その時間T[i]と1つ前のイベントの時間T[i-1]との差 $\Delta T[i] = (T[i] - T[i-1])$ を計算し（ステップS11）、ノートテーブルの所定の欄に書き込む（但し、 $i = 1 \sim NA$, $T[0] = 0$ ）。すなわち、各ノートイベント間の相対時間が求まる。

50 【0034】ここで、SMFにおいて、Δタイムは1拍の何分の1かを基本単位にした可変長符号で表され、値が小さいほど必要なバイト数は少なくて済む。例えば、値が127以下であれば1バイトでよいが、値が128

以上16383以下であれば、2バイト必要となる。 Δ タイムの基本単位が細かいほど音楽的な表現力は高いと言えるが、それによって必要なバイト数も増える。一方、実際に楽曲に使われている Δ タイムを調べると、基本単位の1刻み(1 tick)まで使っていないことが多く、したがって、 $\Delta T[i]$ の値が必要以上の容量を使って記録されていることが多い。

【0035】そこで、実際に使用されている時間精度を求める為に、ノートテーブルに登録されている全ての相対時間 $\Delta T[i]$ に対する最大公約数 ΔTs を算出する(ステップS12)。この場合、最大公約数を求めるのが困難な場合は、適当な公約数を最大公約数 ΔTs とする。次いで、 $\Delta T[i]$ をSMFと同様の可変長符号で出力する(ステップS13~S17)。ただし、 $\Delta T[i]$ の値を可変長符号にする際に、 $\Delta T[i]$ を ΔTs で除算した値 $\Delta Ta[i]$ を符号化する(ステップS15)。したがって、ノート Δ 符号は図9のように最大公約数 ΔTs と、NA個の値 $\Delta Ta[i]$ ($i=1\sim NA$)により構成される。

【0036】ここで、本演奏情報圧縮装置により圧縮された符号を伸長する際には、 $\Delta Ta[i]$ の値を読み取り ΔTs をかけ合わせることで元の $\Delta T[i]$ が復元され、したがって、SMFの持つ音楽的な表現力を失うことなくデータ量を削減することができる。例えば Δ タイムの基本単位を一般的に良く使われる480分の1拍とし、 $\Delta Ts=10$ である場合を例にすると、SMFでは1拍の長さ $\Delta T=480$ や1/2拍の長さ $\Delta T=240$ を表現するのに各々2バイト必要である。一方、本発明では ΔTs で除算して表現することにより $\Delta T=48$ あるいは $\Delta T=24$ を表現すればよいことになり、各々1バイトの使用で済む。また、1拍あるいは1/2拍に相当する Δ タイムは使用頻度が高いので、これらが1つの Δ タイムにつき1バイトずつ削減されれば、楽曲全体で相当の容量を削減することができる。

【0037】-コントローラ Δ 符号生成手段14は、処理対象がノートテーブルではなくコントローラテーブルで

$$\text{num}[i] = f(\text{num}[i-1], \text{num}[i-2], \dots, \text{num}[i-S]) + \alpha[i]$$

但し、イベントの数をNAとして、

$$i = (S+1), (S+2), \dots, NA$$

ここで、関数 $f()$ には種々のものが考えられるが、なるべく同じ値の $\alpha[i]$ が繰り返して出現するようなものを選ぶことにより、2次符号生成手段4で効率の良い

$$\text{num}[i] = \text{num}[i-1] + \alpha[i] \quad \dots (2)$$

但し、イベントの数をNAとして、

$$i = 2, 3, \dots, NA$$

ここで、通常の楽曲においては、コード(和音)のルート音(根音)の平行移動量と同じ音程だけ移動したメロディーラインが存在することが多い。例えば「C」コードの小節で「ド、ド、ミ、ソ、ミ」というメロディーラ

あるという点以外はノート Δ 符号生成手段13と全く同じ処理であり、生成されるコントローラ符号の構成も符号の数がNA個からNB個に変わる以外は、図9に示すノート Δ 符号と同じである。

【0038】デュレーション符号生成手段15はノート Δ 符号生成手段13とほぼ同じであり、図10に従って処理を行う。まず、ノートテーブルに登録された各デュレーションの値の最大公約数 Ds を算出する(ステップS21)。この場合、最大公約数を求めるのが困難な場合は、適当な公約数を最大公約数 Ds とする。デュレーション符号は図11に示すように最大公約数 Ds と、NA個の値 $Da[i]$ ($i=1\sim NA$)により構成され、最大公約数 Ds に続き、各デュレーションの値 $D[i]$ を Ds で割った値 $Da[i]$ を可変長符号として出力する(ステップS22~S26)。ここで、デュレーションは前述したように、SMFのノートオンイベントとノートオフイベントの間の Δ タイムに相当するので、ノート Δ 符号生成手段13において説明したのと同様な理由により、SMFに比べデータ量が削減される。

【0039】ノートナンバ符号生成手段16では、ノートテーブルに登録されているノートナンバに対し、以下の処理を施すことによりノートナンバ符号を生成する。ここで、あるノートナンバ $\text{num}[i]$ を(1)式のようにそれ以前のS個のノートナンバ $\text{num}[i-1]$ 、 $\text{num}[i-2]$ 、 \dots 、 $\text{num}[i-S]$ を変数とする関数 $f()$ と残差 $\alpha[i]$ で表す(ただし、 $\text{num}[i-1]$ は $\text{num}[i]$ の1つ前のノートナンバ、 $\text{num}[i-2]$ は $\text{num}[i]$ の2つ前のノートナンバを表す)。

【0040】ノートナンバ符号は図12に示すように、 $i \leq S$ のイベントに対するノートナンバと、 $i > S$ のイベントに対して残差 $\alpha[i]$ を時間順に並べたものにより構成される。したがって、圧縮時と伸長時で同じ関数 $f()$ を使えば、残差 $\alpha[i]$ から $\text{num}[i]$ を復元することができる。

$$\dots (1)$$

圧縮が可能となる。一例として(2)式のような関数を使った場合の効果を説明する。これは $S=1$ であり、1つ前のノートナンバとの差分を $\alpha[i]$ とすることを意味する。ただし $i=1$ の場合はノートナンバそのものをノートナンバ符号として出力する。

インがある場合に、ルート音が2度高い「D」コードの小節において「レ、レ、#ファ、ラ、レ」というように、最初のメロディーラインを2度上げたメロディーラインが存在することが多い。

【0041】この各々のメロディーラインをSMFのノートナンバそのもので表すと、「60、60、64、6

7, 60]、[62, 62, 66, 69, 62]となり、この2つに共通のデータパターンは全くない。しかし前述の $\alpha[i]$ で表現すると、どちらのメロディーラインもその2音目以降は「0, 4, 3, -7」となり同一のパターンとなる。このようにSMFでは全く異なる2つのデータパターンを、本手法により同一のデータパターンに変換することができる。

【0042】LZ法では、同一のデータパターンが多いほど圧縮率が高くなるので、このようなノートナンバの表現方法により圧縮率が高くなることは明らかである。なお(1)式で $S=0$ とすると、

$$\text{num}[i] = \alpha[i]$$

となり、ノートナンバそのものを符号化することになる。また関数 $f()$ を何種類か用意しておき、最も適切な関数を選択して符号化するとともに、どの関数を使用したかという情報を合わせて符号化してもよい。

【0043】ベロシティ符号生成手段17もノートナン

$$\text{vel}[i] = g(\text{vel}[i-1], \text{vel}[i-2], \dots, \text{vel}[i-T]) + \beta[i] \quad (3)$$

但し、イベントの数をNAとして、

$$i = (T+1), (T+2), \dots, NA$$

次に、コントローラ符号生成手段18について説明する。コントローラ符号は図14に示すように、図6に示すコントローラテーブルに登録されたイベントの情報を時間順に並べたものである。各コントローラ符号は、イベントの種類を表すフラグFとパラメータ(データバイト)で構成される。パラメータの個数はイベントの種類により異なる。イベントの種類は大きく分けて「通常イベント」と「連続イベント」の2つのタイプがある。フラグFの最上位ビットが「1」、パラメータの最上位ビットが「0」に設定されているので、SMFと同様のランニングステータス表現(前のイベントと同じ種類のイベントの場合にフラグFを省略すること)が可能になっている。

【0045】ここで、SMFではイベントの種類をあらわすのに1バイトのMIDIステータスが使われている。一般的に使用される値は、8n(hex)、9n(hex)、An(hex)、Bn(hex)、Cn(hex)、Dn(hex)、En(hex)、F0(hex)、FF(hex)のいずれかである(ただし、 $n=0 \sim F(\text{hex})$ 、nはチャンネル番号である)。「通常イベント」は、上記MIDIステータスからノートオン8n(hex)とノートオフ9n(hex)を除いたものであるが、本発明では前述したようにチャンネル番号を表現する必要が無いので、「通常イベント」のフラグの種類は7種類となる。従ってMIDIステータスに比べフラグFは同じ値になる確率が高く、LZ法を用いた場合の圧縮率が高まる。「通常イベント」の符号は、フラグFの後に、SMFのMIDIステータス1バイトを除いたデータバイトを並べたものである。

バ符号生成手段16と同様である。ノートテーブルに登録されたある音符のベロシティ $\text{vel}[i]$ を(3)式のように、それ以前に出現したT個の音符のベロシティ $\text{vel}[i-1], \text{vel}[i-2], \dots, \text{vel}[i-T]$

を変数とする関数 $g()$ と残差 $\beta[i]$ で表す(ただし、 $\text{vel}[i-1]$ は $\text{vel}[i]$ の1つ前のベロシティ、 $\text{vel}[i-2]$ は $\text{vel}[i]$ の2つ前のベロシティ)。

【0044】ベロシティ符号は図13に示すように、 $i \leq T$ のイベントに対するベロシティと、 $i > T$ のイベントに対して残差 $\beta[i]$ を時間順に並べたものにより構成される。したがって、圧縮時と伸長時で同じ関数 $g()$ を使えば、残差 $\beta[i]$ から $\text{vel}[i]$ を復元でき、また、関数 $g()$ を適当に選ぶことにより、同じデータパターンの $\beta[i]$ が繰り返して出現することになり、LZ法を用いた場合の圧縮率を改善することができる。

20 【0046】また、SMFでは、特定の種類のイベントが一定数以上連続して出現し、各々のイベントのパラメータ値(データバイト)がほぼ一定の規則で変化する部分が存在することが多い。例えば「ピッチホイールチェンジ」イベントが使われている部分である。このイベントは、音符の音程を微妙に変えて音楽的な表現力を高める為のものであり、その性質上パラメータ値の異なる複数イベントが連続して使われることが多い。このようなイベントを「連続イベント」と呼び、このような部分を「連続イベントブロック」と呼ぶ。

30 【0047】以下の説明では、「連続イベント」の一例として「ピッチホイールチェンジ」を取りあげるが、これに限定されるものではない。SMFの「連続イベントブロック」の一例を図15に示す。この場合、各イベントのパラメータ値が異なる為、SMFの同一データパターンの長さは、 $(\Delta \text{タイム} \cdot 1 \text{ バイト} + \text{ステータス} \cdot 1 \text{ バイト})$ の計2バイトであり、この程度の長さではLZ法による圧縮の効果はほとんど得られない。

40 【0048】そこで、コントローラテーブルの中で「ピッチホイールチェンジ」が一定数以上連続して出現し、パラメータ値がほぼ一定の規則で変化する領域に対し、以下の処理を施すことによりコントローラ符号を生成する。まず、「ピッチホイールチェンジ」の数が一定数に満たないような場合は、前述した「通常イベント」として符号化する。そして、(4)式に示すように、連続イベントブロック内のイベントのパラメータ $p[i]$ をそれ以前に出現したU個のイベントのパラメータ値 $p[i-1], p[i-2], \dots, p[i-U]$ を変数とする関数 $h()$ と残差 $\gamma[i]$ で表す(ただし、 $p[i-1]$ は $p[i]$ の1つ前のパラメータ値、 $p[i-2]$ は $p[i]$ の2つ前のイベントのパラメータ

値)。

【0049】連続イベントの符号の構成は図16に示すように、ピッチホイールチェンジが連続していることを意味するフラグFに続き、1番目からU番目までのイベ

$$p[i] = h(p[i-1], p[i-2], \dots, p[i-U]) + \gamma[i]$$

(4)

ただし、連続イベントブロックのイベント数をNCとして $i = (U+1), (U+2), \dots, NC$

関数 $h()$ には種々のものが考えられるが、なるべく同じ値の $\gamma[i]$ が繰り返して出現するようなものを選ぶことにより、2次符号生成手段4において効率の良い圧縮が可能となる。一例として(5)式のような関数を使った場合の効果を説明する。これは $U=1$ であり、1つ前のパラメータとの差分をとることを意味する。

$$p[i] = p[i-1] + \gamma[i] \quad (5)$$

ただし、連続イベントブロックのイベント数をNCとして

$i = i = 2, 3, \dots, NC$

$$p[i] = e(p[i-1], p[i-2], \dots, p[i-U], t[i], t[i-1], \dots, t[i-U]) + \gamma[i] \quad (6)$$

ただし、連続イベントブロックのイベント数をNCとして

$i = (U+1), (U+2), \dots, NC$

符号配置手段19では、上記の各符号を図18のような領域に配置して1次符号3を生成する。ヘッダは各符号の開始アドレスや長さといった管理情報と前述したチャネルマップを含んでいる。すでに説明したように各符号はSMFに比べ、同一データの出現回数が多く、同一データパターンの長さも長いという性質を持っているが、さらに同一データパターンが近い距離で出現するように工夫をしている。まず、同じ種類の符号内で、同じデータ列が出現する確率が高いので、トラック順に同一種類の符号を配置している。また、ノートΔ符号とコントローラΔ符号とデュレーション符号は、全て時間情報であり、性質の異なるノートナンバ符号やベロシティ符号よりも同じデータ列が出現する確率が高いので、これらの距離が近くなるように配置している。

【0050】次に、図19で示した例に戻って、同一データパターンの長さがどの程度改善されるか具体的に検討する。ここで、各々のメロディは、50個のノートオンイベントと50個のノートオフイベントで構成されており、全てのΔタイムは1バイトであるとし、全てのイベントは3バイトであると仮定すると、各々のメロディのノートナンバは前述したように全て同じである。

【0051】SMFにおいて、各々のメロディのデータ量は、

$$(1+3) \times 50 \times 2 = 400 \text{ バイト}$$

である。各々のメロディのΔタイムとベロシティが全て同じならば、同一データパターンの長さは400バイトになる。しかし両メロディ間の全てのΔタイムとノート

ントに対してはパラメータの値そのものである。そして、 $(U+1)$ 番目以降のイベントでは、 $\gamma[i]$ を時間順に並べたものである。

この方法によれば、図15に示す領域は図17のようなコントローラ符号に変換される。この場合、2番目以降のイベントのデータが全て同一の「1」になる為、LZ法による圧縮率が高まる。また、コントローラ符号にはΔタイムが含まれないので、Δタイムがイベント毎に異なる場合でも、LZ法における圧縮率低下の影響が少ない。また場合によっては、(4)式の代わりに、イベントの時間情報も変数に使った(6)式のような関数 $e()$ を使っても良い。ただし、 $t[i]$ はパラメータを求めるイベントの時間、 $t[i-1]$ はその1つ前のイベントの時間である。

オンのベロシティが異なっているとすると、SMFで同一データパターンの最大長は、ノートオフステータス、ノートナンバ、ベロシティの並びの3バイトである。この程度ではLZ法の圧縮はほとんど効果がない。

【0052】一方、本発明では、Δタイム、ノートナンバ、ベロシティを分離して符号化しているので、少なくともノートナンバ符号の中で50バイトの長さの同一データパターンが出現する。また前述したようにSMFのベロシティが全く異なる場合でも、ベロシティ符号の中では同一データパターンが出現することが多い。従ってLZ法による圧縮率は明らかに改善される。以上の説明から分かるように1次符号3は、SMFの持つ音楽的な情報量を全く落とすことなくデータ量が削減されていると同時に、SMFに比べ同一のデータパターンの長さが長く、出現回数も多く、しかもそれらが近い距離で出現するよう性質を持っているので、2次符号生成手段4において効果的な圧縮を行うことができる。また、この1次符号3そのものも、かなり圧縮されたデータ量となっているので、この1次符号3を直接出力するにしてもよい。

【0053】2次符号生成手段4においては、1次符号生成手段2の出力3に対して、LZ法による圧縮を行う。LZ法は、gzip、LHAといった圧縮プログラムで広く使われている手法である。これは入力データの中から、同一のデータパターンを捜し、もし存在すれば、(過去に出現したパターンへの距離、パターンの長さ)という情報に置き換えて表現することでデータ量を削減する。例えば、

"ABCDEABCDEF"

というデータを、"ABCDE" が繰り返してであるの

で、

”ABCDE (5, 5) F”

という情報に置き換える。なお、圧縮符号 (5, 5) は 5 文字戻って 5 文字コピーすることを表す。

【0054】処理の概要は次のようになる。2 次符号 5 の生成は、処理位置を 1 次符号 3 の先頭から順次移動させて行う。処理位置のデータパターンが、それ以前の一定の範囲内のデータパターンと一致する場合は、処理位置からそのデータパターンまでの距離と、一致したデータパターンの長さを 2 次符号 5 として出力し、処理位置を 2 つ目のデータパターンの終わりに移動させ、処理を続ける。処理位置のデータパターンが、それ以前の一定の範囲内のデータパターンと一致しなければ、1 次符号 3 をコピーして 2 次符号 5 として出力する。

【0055】以上の説明から明らかなように、2 つの同一のデータ領域が大きいほど、圧縮率は高くなる。また同一のデータ領域の距離は一定範囲内である必要がある。前述したように楽曲の中では、同じようなメロディーが繰り返し使われるが、SMF のままでそれらのデータを比較すると、完全に同一のデータ列の繰り返しであることは少なく、むしろノートナンバは同じであるがベロシティは異なるといったように、どこか一部分異なっていることが多い。

【0056】一方、本発明では、性質の同じデータを独立した領域にまとめると同時に、各領域で同一のデータがなるべく多く出現するような処理を行ない、さらに性質の近い領域どうしをなるべく近くに配置することにより、LZ 法の圧縮率が高まるので、最終的な 2 次符号 5 は十分容量の小さなものになる。なお、以上詳述したフォーマット並びに処理手順は一例であり、その主旨を逸脱しない範囲において種々の変更を加えることができる。また、演奏情報として SMF を例に取ったが、SMF に限らずこれに類似の演奏情報に対して本発明を適用してデータ容量を効率よく削減することができる。

【0057】次に、図 20～図 28 を参照して上記の 1 次符号 3 又は 2 次符号 5 を復号するための演奏情報復号装置について説明する。図 20 は演奏情報復号装置を示すブロック図、図 21 は図 20 の 2 次符号復号手段の処理を説明するためのフローチャート、図 22 は図 20 の 1 次符号復号手段の処理を説明するためのフローチャート、図 23 は図 22 のトラック復号処理を詳しく説明するためのフローチャート、図 24 は図 23 のノートイベント復号処理を詳しく説明するためのフローチャート、図 25 は図 24 のノートイベント復号処理により復元されたノートオンイベントを示す説明図、図 26 は図 24 のノートイベント復号処理により復元されたノートオフイベントを示す説明図、図 27 は図 23 のコントローライベント復号処理を詳しく説明するためのフローチャート、図 28 は図 27 の処理により復元されたコントローライベントを示す説明図である。

【0058】図 20 では圧縮処理とは逆に、LZ 法で圧縮された入力データ 21 が 2 次符号復号手段 23 により音程と、音の強さと、音の長さその他の情報に分離された 1 次符号 3 に復号され、次いで 1 次符号復号手段 24 により元の音符（出力データ 25）に復元される。制御手段 26 はスイッチ 22 により、入力データ 21 が図 1 に示す 2 次符号 5 である場合に 2 次符号復号処理に続き 1 次符号復号処理を行うように制御し、入力データ 21 が図 1 に示す 1 次符号 3 である場合に 1 次符号復号処理のみを行うように制御する。

【0059】ここで、2 次符号 5 であるか又は 1 次符号 3 であるかの判定は、キーボード、マウス、ディスプレイ等の図示しない入出力装置を使用してオペレータが指定してもよいし、圧縮された情報に対して符号化方法の種類を示す情報を符号化時に付加し、復号時にこの情報を自動的に判別するようにしてもよい。

【0060】次に、図 21 を参照して 2 次符号復号手段 23 の復号処理を説明する。入力データ 11（2 次符号 5）を先頭から読み込み（ステップ S101）、次いで圧縮データの部分であるかすなわち ABCDE (5, 5) の「ABCDE」の部分（＝非圧縮データ）であるか又は「(5, 5)」の部分（＝圧縮データ）であるかを判定する（ステップ S102）。

【0061】そして、圧縮データの部分である場合には過去に出現した同一パターンを参照してそれをコピーして出力し（ステップ S103）、他方、非圧縮データの部分である場合にはそのまま出力する（ステップ S104）。以下、入力データ 11 を全て復号するまでこの処理を繰り返すと（ステップ S105→S101）、図 18 に示すような配置の 1 次符号 3 が復元される。

【0062】次に、図 22 及び図 18 に示す 1 次符号 3 を参照して 1 次符号復号手段 24 の復号処理を説明する。まず、1 次符号 3 のヘッダを読み込む（ステップ S111）。ヘッダには総トラック数 N、ノート Δ 符号からコントロール符号までの各符号領域の先頭アドレス、チャンネルマップ、時間分解能等の情報が符号化の際に記録されているので、このヘッダ情報に基づいて SMF のヘッダを作成して出力する（ステップ S112）。

【0063】次にトラック番号 i を「1」にセットし（ステップ S113）、図 23 に詳しく示すトラック復号処理を行う（ステップ S114）。次いでトラック番号 i が総トラック数 N より小さいか否かをチェックし（ステップ S115）、もし小さければトラック番号 i を 1 つインクリメントし（ステップ S116）、ステップ S114 に戻ってトラック復号処理を繰り返す。そして、ステップ S115 においてトラック番号 i が総トラック数 N より小さくない場合にこの 1 次符号復号処理を終了する。

【0064】図 23 に詳しく示すトラック復号処理では、まず、処理で使用する変数を初期化する（ステップ

S121)。具体的は、処理中のノートイベントの番号を示す変数jを「1」にセットし、処理中のコントローライベントの番号を示す変数kを「1」にセットし、ノート終了フラグとコントローラ終了フラグをクリアする。ここで、ノート終了フラグは処理トラックの全てのノートイベントの復号が終了したことを示し、コントローラ終了フラグは処理トラックの全てのコントローライベントの復号が終了したことを示す。

【0065】次に処理トラック番号iのノートΔ符号の最大公約数ΔTsnと、コントローラΔ符号の最大公約数ΔTscとデレージョン符号の最大公約数Dsを読み出す(ステップS122)。そして、j番目のノートΔ符号ΔTan[j]とk番目のコントローラΔ符号ΔTac[k]を読み出し、(7)式のように各々最大公約数ΔTsn、ΔTscを乗じてΔTn[j]、ΔTc[k]を算出する(ステップS123)。

$$\begin{aligned}\Delta Tn[j] &= \Delta Tan[j] \times \Delta Tsn \\ \Delta Tc[k] &= \Delta Tac[k] \times \Delta Tsc\end{aligned}\quad (7)$$

さらに、(8)式のようにトラックの先頭を基準とした時刻Tn[j]、Tc[k]に変換する(ステップS124)。

$$\begin{aligned}Tn[j] &= Tn[j-1] + \Delta Tn[j] \\ Tc[k] &= Tc[k-1] + \Delta Tc[k]\end{aligned}\quad (8)$$

ただし、 $Tn[0] = Tc[0] = 0$

なお、ステップS123、S124では、ノート終了フラグがセットされている場合にはΔTn[j]、Tn[j]の算出は行わず、また、コントローラ終了フラグがセットされている場合にはΔTc[k]、Tc[k]の算出は行わない。

【0066】次に、出力すべきノードオフイベントの有無をチェックし(ステップS125)、出力すべきデータが有る場合にはSMFとしてノートオフイベントを出力する(ステップS126)。なお、ステップS125、S126については後述(図24のステップS144)する。次に、復号処理の選択を行う。まず、コン

$$\begin{aligned}num[j] &= f(num[j-1], num[j-2], \dots, num[j-S]) + \alpha[j] \quad (j > S) \\ num[j] &= \alpha[j] \quad (j \leq S)\end{aligned}\quad (9)$$

ただし、Sは関数f()の変数の個数

同様に、j番目のベロシティ符号β[j]を読み取り、式に従ってベロシティvel[j]を算出する(ステップS142)。

$$\begin{aligned}vel[j] &= g(vel[j-1], vel[j-2], \dots, vel[j-T]) + \beta[j] \quad (j > T) \\ vel[j] &= \beta[j] \quad (j \leq T)\end{aligned}\quad (10)$$

ただし、Tは関数g()の変数の個数

次いで、Tn[j]、num[j]、vel[j]を用いて図25に示すようなノートオンイベントを出力する(ステップS143)。なお、SMTのΔタイムΔT

$$\Delta T = Tn[j] - Tb$$

図25に示すノートオンイベントにおけるステータスバ

ローラ終了フラグをチェックし(ステップS127)、セットされている場合には図24に詳しく示すノートイベント復号処理を行う(ステップS128)。

【0067】コントローラ終了フラグがセットされていない場合にはノート終了フラグをチェックし(ステップS129)、セットされている場合には図27に詳しく示すコントローライベント復号処理を行う(ステップS130)。2つのフラグが共にセットされていない場合にはTn[j]とTc[k]を比較し(ステップS131)、Tn[j]が小さければノートイベント復号処理(ステップS128)を、そうでなければコントローライベント復号処理(ステップS130)を行う。

【0068】ノートイベント復号処理の後には、処理トラックNの全てのノートイベントを処理したか否かチェックし(ステップS132)、処理が終了している場合にはノート終了フラグをセットし(ステップS133)、ステップS138に進み、そうでなければ変数jを1つインクリメントし(ステップS134)、ステップS123に戻る。また、コントローライベント復号処理の後には、処理トラックNの全てのコントローライベントを処理したか否かチェックし(ステップS135)、処理が終了している場合にはコントローラ終了フラグをセットし(ステップS136)ステップS138に進み、そうでなければ変数kを1つインクリメントし(ステップS137)、ステップS123に戻る。

【0069】ステップS138ではノート終了フラグとコントローラ終了フラグの両方がセットされているか否かチェックし、両方がセットされている場合にはこのトラック復号処理を終了し、そうでなければステップS123に戻ってこのトラック復号処理を繰り返す。

【0070】図24に詳しく示すノートイベント復号処理では、まず、j番目のノートナンバ符号α[j]を読み取り、圧縮処理において使用した関数f()を用いて(9)式に従ってノートナンバnum[j]を算出する(ステップS141)。

$$num[j] = f(num[j-1], num[j-2], \dots, num[j-S]) + \alpha[j]$$

$$(j \leq S)$$

$$(9)$$

式に従ってベロシティvel[j]を算出する(ステップS142)。

$$vel[j] = g(vel[j-1], vel[j-2], \dots, vel[j-T]) + \beta[j]$$

$$(j \leq T)$$

$$(10)$$

は、Tn[j]の直前に出力したイベントの時刻Tbを使って式(11)に従って求め、出力する。

$$(11)$$

50 イトの上位4ビットはノートオン「9(hex)」を表

し、下位4ビットはチャネルマップから得られる番号が続く。このステータスバイトの後にはノートナンバとベロシティの各バイトが続く。

【0071】次にノートオフイベントの登録を行う（ステップS144）。具体的にはデュレーション符号Da_j [j]を読み取って、(12)式に従いノートオフイベ

$$T_{off}[j] = Da[j] \times Ds + Tn[j] \quad (12)$$

前述した図23のステップS125においては、Tn[j]とTc[k]の内の値が小さいほうTmをノートオフキューの先頭のToff[n] (n=1~エン트리総数N)から順に比較する。Toff[n]<Tmであるエン

トリがあればステップS126に進み、ノートオフイベントを出力する。ステップS126では、前述したノートオフイベントをSMFとして出力する。

【0072】次に図27を参照してコントローライベント復号処理を詳しく説明する。この処理では図28に示すようにΔタイム、ステータス及びパラメータより成る

コントローライベントが復元され、先ず、Tc[k]と直前に出力したイベントの時刻Tbを使って(13)式に従ってSMTのΔタイムΔTを求め、出力する（ステップS151）。

$$\Delta T = Tc[k] - Tb \quad (13)$$

次にコントローラ符号領域からイベントの種類を表すイ

イベントフラグF[k]を読み取り、F[k]が「通常イ

イベント」であるか、「連続イベント」であるか又は「ランニングステータス」であるかを判定する（ステップS152）。ここで、連続イベントブロック内では、図示

省略16に示すように、2番目以降のイベントはイベントフラグが省略された「ランニングステータス」状態で

$$p[m] = h(p[m-1], p[m-2], \dots, p[m-U] + \gamma[m] \quad (m > U) \\ p[m] = \gamma[m] \quad (m \leq U)$$

ただし、Uは関数h()の変数の個数

F[k]が「ランニングステータス」である場合には、変数mの値をチェックし（ステップS159）、mが「0」より大きければmを1つインクリメントし（ステップS160）、「連続イベント」側のステップS157に進む。他方、mが「0」であれば「通常イベント」側のステップS154に進む。

【0075】次に本発明の、SMFを時間軸上で2つ以上のブロックに分割し、少なくとも最初のブロックは前記1次圧縮及び／又は前記2次圧縮を行わないことを特徴とする演奏情報圧縮装置について説明する。

【0076】図29に本発明による圧縮を行ったときの第1の実施例を示す。従来の方法では、元の演奏情報はそのまま1曲ごとに圧縮を行っているが、演奏情報を着信メロディなどで使用する場合には、電話受信後、直ちに着信メロディが再生されなければならない、圧縮されたファイルを解凍するために端末側の処理速度を高速化する必要がある。そこで、図29のように、演奏情報のフ

ントの時刻Toffを算出し、この時刻Toffとノートナンバnum[j]を図26に示すようなノートオフキューに登録する。このノートオフキューでは、使用されているエントリの数を保持するとともに、ノートオフ時刻Toffが先頭から小さい順に並ぶように管理される。

記録されている。

【0073】F[k]が「通常イベント」である場合には、処理イベントの連続イベントブロック内における順番を示す変数mを「0」にリセットし（ステップS153）、次いでチャネルマップを参照してSMFのステータスバイトを作成して出力する（ステップS154）。さらにイベントの種類に応じて必要なバイト数をコントローラ符号領域から読み出し、この読み出した値がSMFのパラメータ（データバイト）であるのでこれを出力する（ステップS155）。

【0074】F[k]が「連続イベント」である場合には、連続イベントブロック内における順番を示す変数mを「1」にセットし（ステップS156）、次いでチャネルマップを参照してSMFのステータスバイトを作成して出力する（ステップS157）。なお、m≥2の場合のステータスバイトはmが「1」の場合のステータスバイトを利用する。そして、この「連続イベント」の場合には、パラメータ符号γ[m]を読み出し、圧縮処理と同じ関数h()を使い、(14)式に従ってSMFのパラメータp[m]を作成し、出力する（ステップS158）。

$$(14)$$

ファイルを時系列上にブロック1とブロック2の2つのブロックに分割し、それぞれのブロック単位で圧縮を行う。このとき、ブロック1はできるだけ端末側で即座にデータを解凍できるように1次符号化のみの圧縮を施し、2次符号化は行わない。ブロック2は全体のメモリー容量や配信ファイルサイズを小さくするために、通常の1次符号化ならびに2次符号化による圧縮を行う。このようにして、1次符号化のみ施された圧縮ブロック1と1次符号化及び2次符号化が施された圧縮ブロック2とを一つのファイルとして配信伝送し、端末側のメモリーに記録させる。

【0077】ブロック1は、例えば最初の5秒分など、ある程度短めにしておき、電話に対して着信があったとき、まず、圧縮されたブロック1から解凍し、ブロック1に該当する部分の着信メロディを再生しながら、同時に圧縮されたブロック2を解凍する。そのため、ブロック1はブロック2の解凍を行うための十分な時間があ

ば、可能な限り短い方がよい。これは圧縮されたブロック 1 を解凍する作業時間を短縮する点においても重要である。

【0078】図 35 に第 1 実施例のエンコードのフローを示す。まず、SMF を取りこみ（ステップ 401）、ステップ 402 にて SMF を時間軸上で 2 分割し、最初の 5 秒分をブロック 1、残りの部分をブロック 2 とする。そして、ステップ 403 でブロック 1 を 1 次符号化、これによりステップ 404 にてブロック 1 を圧縮した圧縮ブロック 1 を生成する。そして、ステップ 405 10 にてブロック 2 を 1 次符号化及び 2 次符号化し、ステップ 406 にて圧縮ブロック 2 を生成する。更に、圧縮ブロック 1 及び圧縮ブロック 2 を一つにまとめて（ステップ 407）、エンコードを終了する。

【0079】次に再生時の動作を図 31 及び図 32 を参照しながら説明する。電話の着信が発生したとき、楽曲取り込み手段 134 により、発信相手に対応する楽曲ファイルを情報記憶手段 135 に取り込む。既に情報記憶手段 135 に楽曲ファイルが収録されている場合には、直接ここから解凍を行ってもよい。まず圧縮ブロック 1 20 の解凍を始める。この際、圧縮ブロック 1 は 1 次符号化のみ施されているので、図 31 の 1 次符号復号手段 131 によって復号を行い、直ちに演奏情報の再生を行う。そして演奏情報の再生を行いながら、圧縮ブロック 2 の解凍を始める。圧縮ブロック 2 は図 31 の 2 次符号復号手段 132 によって、2 次復号化が行われ、次に 1 次符号復号手段 131 によって 1 次復号化が行われる。ブロック 1 のファイルサイズは、この 2 次復号化が終了し、ブロック 2 の演奏情報が再生可能となるタイミングまでに時間的な猶予が持てる程度の大きさが好ましい。また、演奏情報はテンポをベースに記述されている場合があるので、再生テンポが端末側で最大（最も早い）となるように設定されている場合にも間に合うようにしなければならない。展開した楽曲情報は情報記憶手段 135 に記録される。そして楽曲再生手段 136 により、情報記憶手段 135 に展開された演奏情報が再生される。なお、上述した実施例では発信相手に対応する楽曲ファイルを選んでいますが、発信相手の全てに同じ着信メロディを使用し、電話の使用者の好みに応じて楽曲を切り替えるようにしてもよい。

【0080】次に、本発明による圧縮を行ったときの第 2 の実施例について図 30 を参照して説明する。これは、第 1 の実施例よりも端末の処理が遅い場合に、全体の圧縮率はやや下がるが、ブロック 1 の部分については元のデータをそのまま用いるものである。図 36 に第 2 実施例のエンコードのフローを示す。なお、ブロック 1 の 1 次符号化を行わない点を除いて上述した第 1 の実施例と同じであるため説明を省略する。また、再生時の動作についても図 31 に示す第 1 の実施例と同様の端末構成で、図 33 に示すように圧縮ブロック 1 の解凍を行わ

ない点異なるものであり説明は省略する。

【0081】次に、本発明による圧縮を行ったときの第 3 の実施例について図 34 を参照して説明する。これは、ある程度端末側の処理速度の高速化が見込まれる場合に、ブロック 1 にも 1 次符号化ならびに 2 次符号化を施すものである。図 37 に第 3 実施例のエンコードのフローを示すがこれについてもブロック 1 に対して 1 次符号化及び 2 次符号化を施す以外には上述した実施例と同様であるため、説明を省略する。次に再生時のフローについてであるも上述した実施例と同様であり、ブロック 1 の部分のみがそれぞれの圧縮内容による違いが生ずるのみであるので、説明は省略する。

【0082】なお、上記は着信メロディについて説明を行ったが、カラオケや BGM であっても、再生指定時に直ちに演奏を開始することができる点で、実用に耐えるものとなるので、これらのような使用形態に適用することも可能である。

【0083】

【発明の効果】以上説明したように本発明によれば、LZ 法により圧縮する前に予め、同一のデータパターンの長さが長く、出現回数が多く且つ近い距離で出現するように、演奏情報を音程と、強さと、長さその他の情報に分離し、各情報をそれぞれ独立した領域に配置した 1 次符号を生成し、この 1 次符号を LZ 法により圧縮するので、演奏情報のデータ量を効率的に圧縮することができるが、このような装置に於いて、SMF を 2 つ以上のブロックに分割し、少なくとも最初のブロックは前記 1 次圧縮及び/または前記 2 次圧縮を行わないことを特徴とする装置を提供することで、圧縮した状態からの再生 30 をすぐに行える端末を提供する事ができる。

【0084】また、1 次符号として演奏情報を音符の音程領域と、音符の強さ領域と、音符の長さ領域とその他の領域の少なくとも 4 つの領域に分離して符号化するので、元の演奏情報のもつ演奏品位を全く失うことなく、従来に比べ大幅にデータ容量が削減でき、したがって、データを保存するのに小容量の記録媒体を用いることができコストを削減することができ、また、通信回線を利用してデータを伝送する場合にもコストを削減することができるとともに、伝送時間を削減することができる。また、大量の演奏情報を扱う通信カラオケや着信メロディなどの音楽データベースや端末のメモリ容量節約に効果 40 が大きい。

【図面の簡単な説明】

【図 1】本発明に係る演奏情報圧縮装置の一例を示すブロック図である。

【図 2】図 1 の 1 次符号生成手段の一例を詳細に示すブロック図である。

【図 3】図 2 のチャンネル分離手段により作成されるチャネルマップを示す説明図である。

【図 4】図 2 の解析手段の処理を説明するためのフロー 50

チャートである。

【図 5】図 2 の解析手段により作成されるノートテーブルを示す説明図である。

【図 6】図 2 の解析手段により作成されるコントローラテーブルを示す説明図である。

【図 7】音符を表現する SMF の Δ タイムと本実施例のデュレーションの関係を示す説明図である。

【図 8】図 2 のノート Δ 符号生成手段の処理を説明するためのフローチャートである。

【図 9】図 2 のノート Δ 符号生成手段により生成されるノート Δ 符号を示す説明図である。

【図 10】図 2 のデュレーション符号生成手段の処理を説明するためのフローチャートである。

【図 11】図 2 のデュレーション符号生成手段により生成されるデュレーション符号を示す説明図である。

【図 12】図 2 のノートナンバ符号生成手段により生成されるノートナンバ符号を示す説明図である。

【図 13】図 2 のベロシティ符号生成手段により生成されるベロシティ符号を示す説明図である。

【図 14】図 2 のコントローラ符号生成手段により生成されるコントローラ符号を示す説明図である。

【図 15】SMF の連続イベントブロックを示す説明図である。

【図 16】本実施例の連続イベントブロックを示す説明図である。

【図 17】図 16 の連続イベントブロックの効果を示す説明図である。

【図 18】図 2 の符号配置手段により並べ替えられた 1 次符号を示す説明図である。

【図 19】SMF のフォーマットを示す説明図である。

【図 20】演奏情報復号装置を示すブロック図である。

【図 21】図 20 の 2 次符号復号手段の処理を説明するためのフローチャートである。

【図 22】図 20 の 1 次符号復号手段の処理を説明するためのフローチャートである。

【図 23】図 22 のトラック復号処理を詳しく説明するためのフローチャートである。

【図 24】図 23 のノートイベント復号処理を詳しく説明するためのフローチャートである。

【図 25】図 24 のノートイベント復号処理により復元されたノートオンイベントを示す説明図である。

【図 26】図 24 のノートイベント復号処理により復元されたノートオフキューを示す説明図である。

【図 25】

ノートオンイベント

Δ タイム	ステータス (0x hex)	ノートナンバ	ベロシティ

【図 27】図 23 のコントローライベント復号処理を詳しく説明するためのフローチャートである。

【図 28】図 27 の処理により復元されたコントローライベントを示す説明図である。

【図 29】本発明に係る演奏情報圧縮装置による圧縮方法の第 1 の実施例を示す図である。

【図 30】本発明に係る演奏情報圧縮装置による圧縮方法の第 2 の実施例を示す図である。

【図 31】本発明に係る演奏情報圧縮装置の構成を示す図である。

【図 32】第 1 の実施例により圧縮された演奏情報の解凍方法を示す図である。

【図 33】第 2 の実施例により圧縮された演奏情報の解凍方法を示す図である。

【図 34】本発明に係る演奏情報圧縮装置による圧縮方法の第 3 の実施例を示す図である。

【図 35】本発明に係る演奏情報圧縮装置による圧縮方法の第 1 の実施例の動作を示す図である。

【図 36】本発明に係る演奏情報圧縮装置による圧縮方法の第 2 の実施例の動作を示す図である。

【図 37】本発明に係る演奏情報圧縮装置による圧縮方法の第 3 の実施例の動作を示す図である。

【符号の説明】

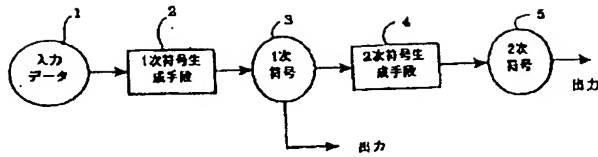
- 1 入力データ
- 2 1 次符号生成手段
- 3 1 次符号
- 4 2 次符号生成手段
- 5 2 次符号
- 11 チャンネル分離手段
- 12 解析手段
- 13 ノート Δ 符号生成手段
- 14 コントローラ Δ 符号生成手段
- 15 デュレーション符号生成手段
- 16 ノートナンバ符号生成手段
- 17 ベロシティ符号生成手段
- 18 コントローラ符号生成手段
- 19 符号配置手段
- 21 入力データ
- 22 スイッチ
- 23 2 次符号復号手段
- 24 1 次符号復号手段
- 25 出力データ
- 26 制御手段

【図 28】

コントローライベント

Δ タイム	ステータス	パラメータ

【図1】

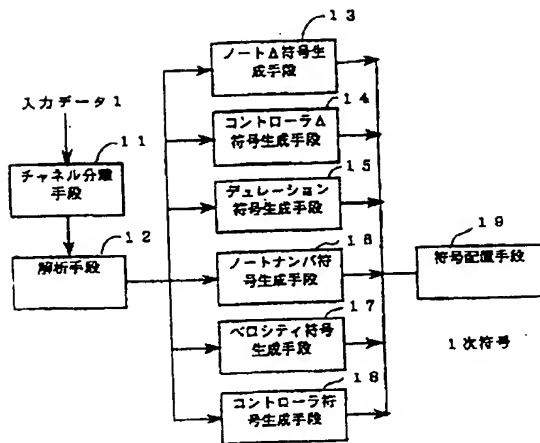


【図3】

チャネルマップ

トラップ	チャネル番号
1	2
2	3
3	5
4	1

【図2】



【図5】

ノートテーブル

	時間	ノートナンバ	ベロシティ	デレション	ノートオフ参照	ΔT
ノート1	480	60	80	240	1	
ノート2	840	62	80		0	
ノートNA						

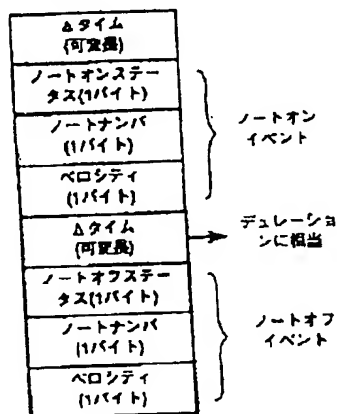
【図6】

コントローラテーブル

	時間	データ
イベント1		
イベント2		
イベントNB		

【図7】

SMF



【図9】

ノートΔ符号

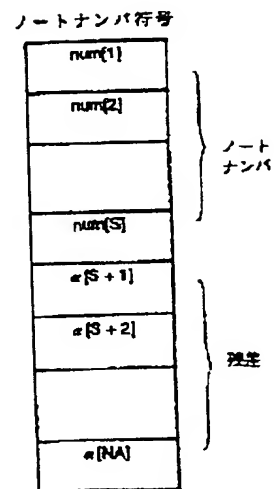
ΔTa
ΔTa[1]
ΔTa[2]
ΔTa[NA]

【図11】

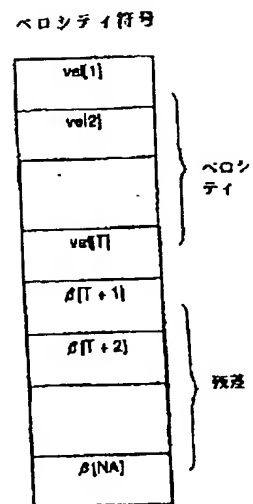
デレション符号

Da
Da[1]
Da[2]
Da[NA]

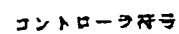
【圖 12】



【图 13】



【图 17】

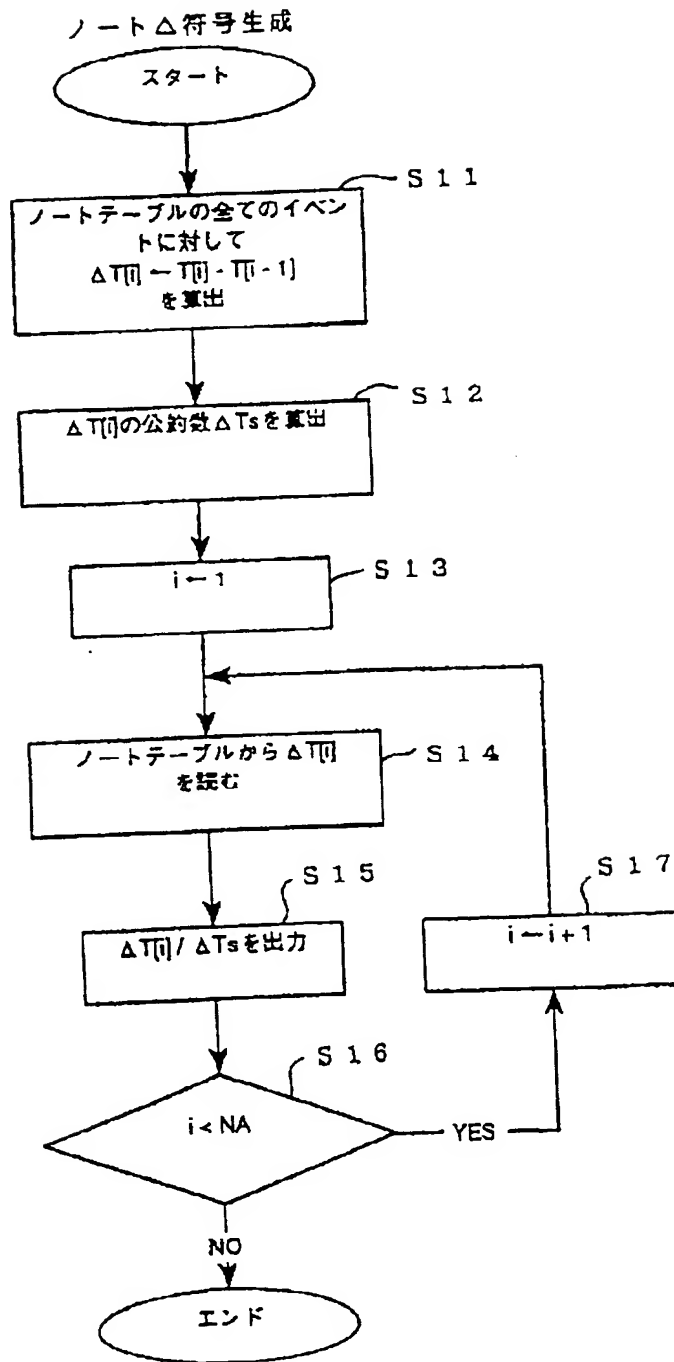


フラグF
初回値 "8192"
残高 "1"
残高 "1"
残高 "1"
残高 "1"
残高 "1"
残高 "1"

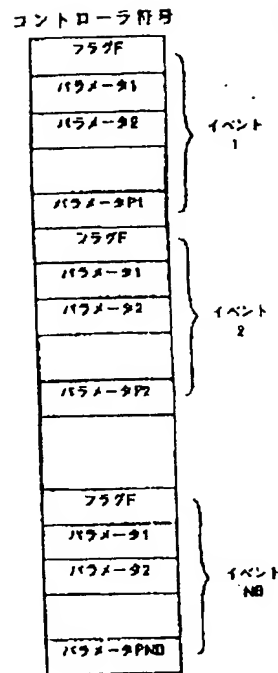
ノートオブキュー

	Yd	ノートナンバ
エントリ1	1000	84
エントリ2	1100	84

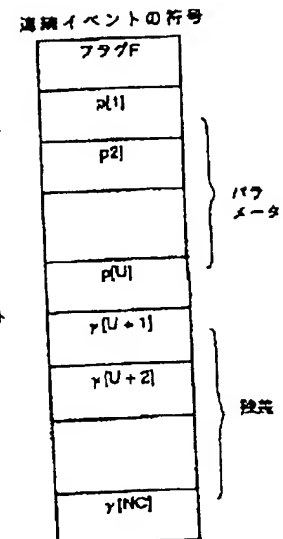
【図8】



【図14】



【図16】

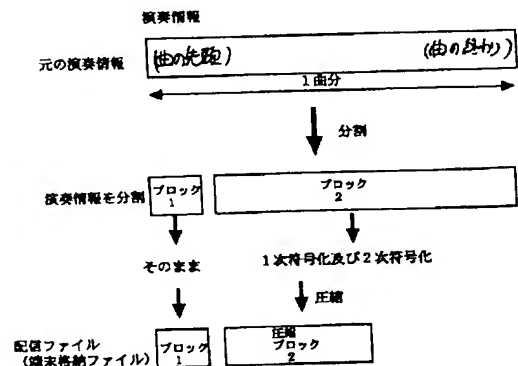


【図15】

連続イベントブロック

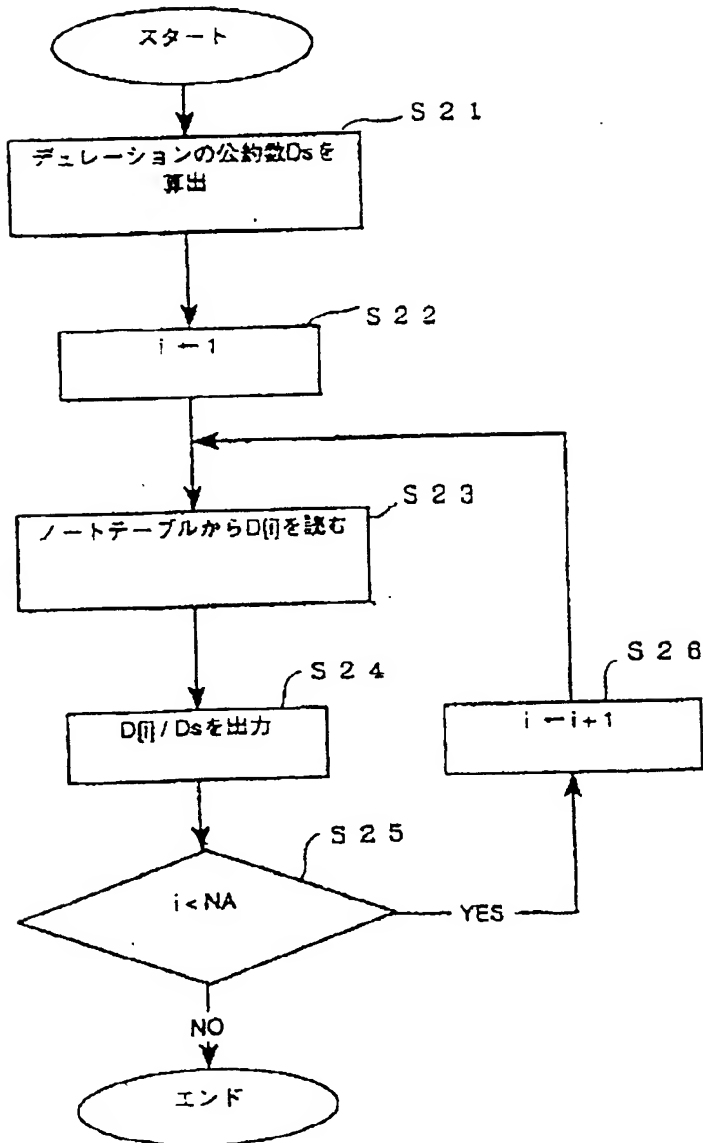
Δタイム	ステータス	パラメータ
10	224(ピッチホイールチェンジ)	8182
20	224(ピッチホイールチェンジ)	8193
20	224(ピッチホイールチェンジ)	8194
10	224(ピッチホイールチェンジ)	8195
30	224(ピッチホイールチェンジ)	8186
10	224(ピッチホイールチェンジ)	8197
20	224(ピッチホイールチェンジ)	8188

【図30】

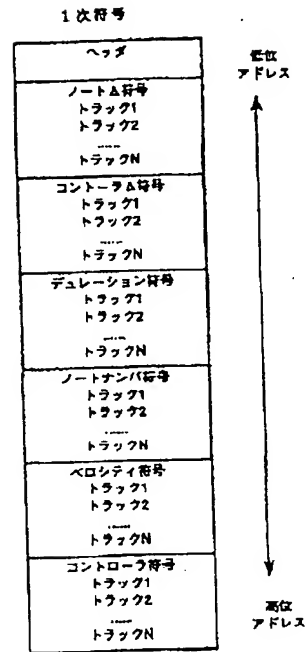


【図10】

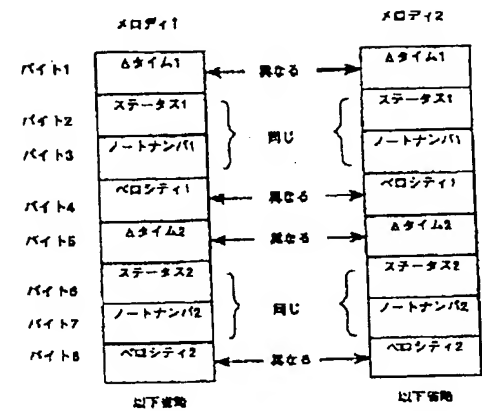
デレージン符号生成



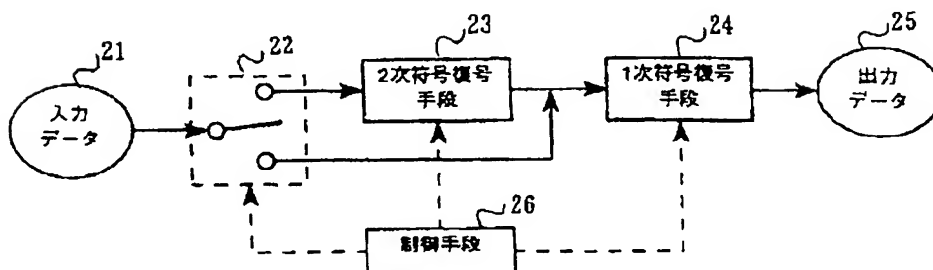
【図18】



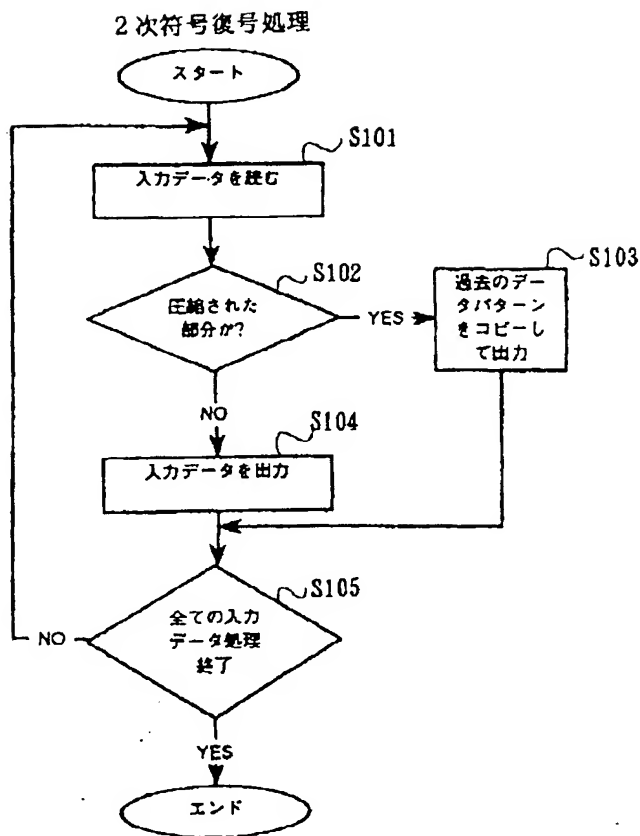
【図19】



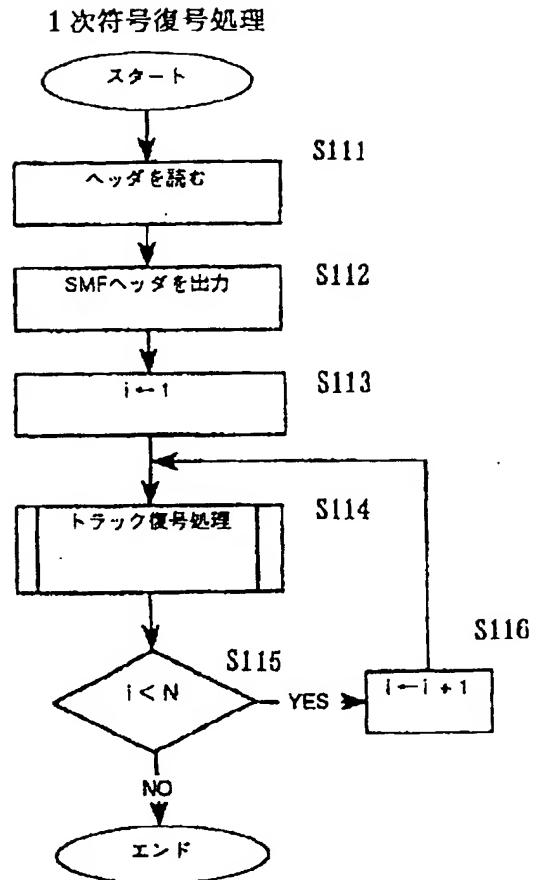
【図20】



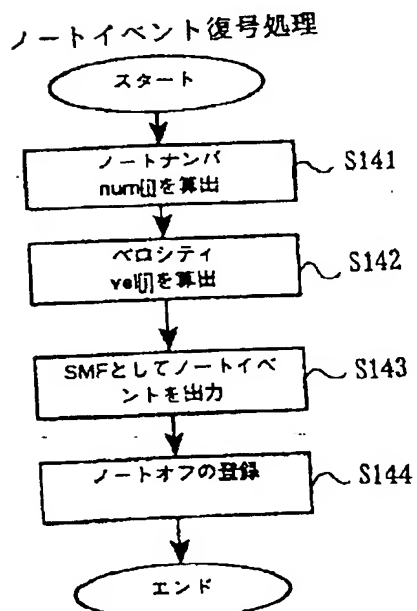
【図21】



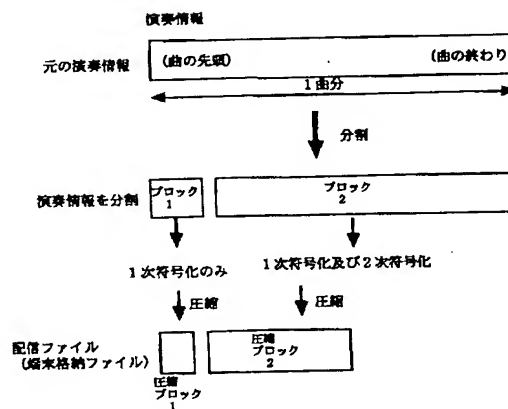
【図22】



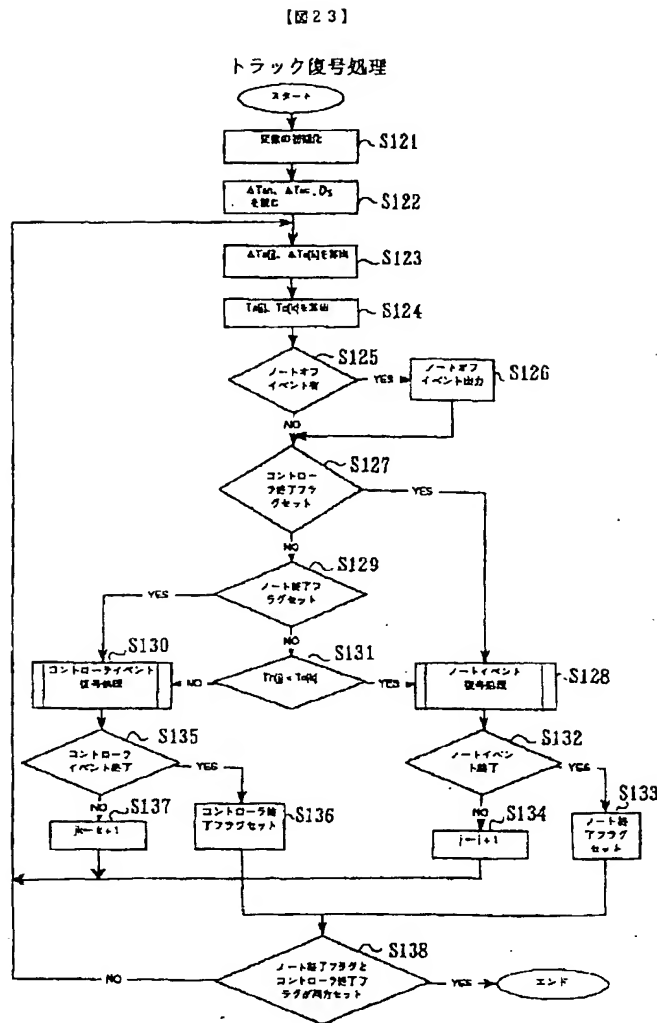
【図24】



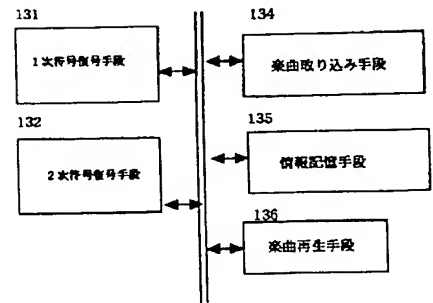
【図29】



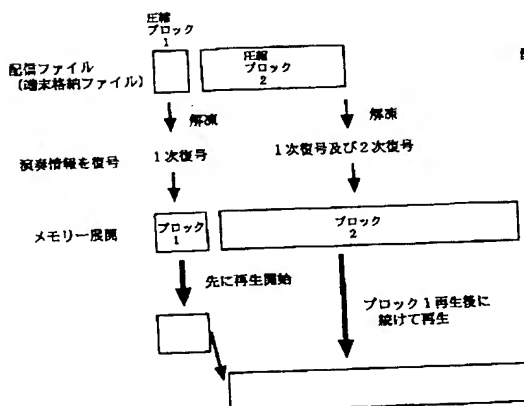
【図 23】



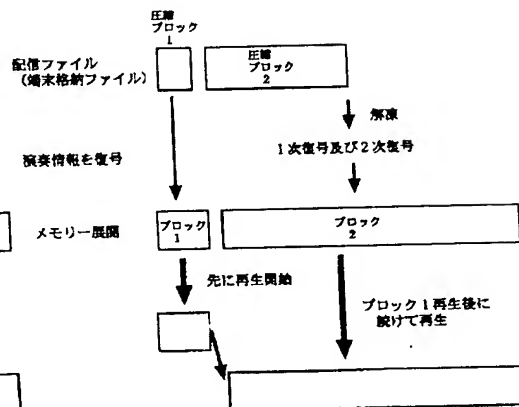
【図 31】



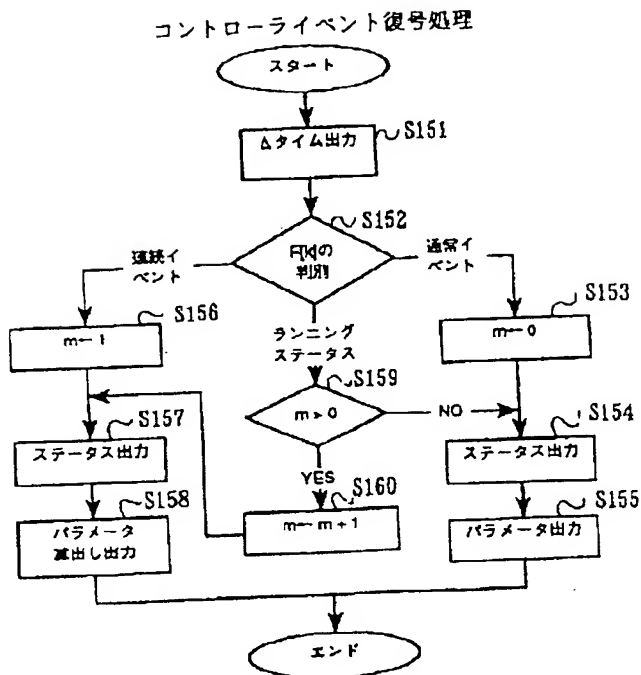
【図 32】



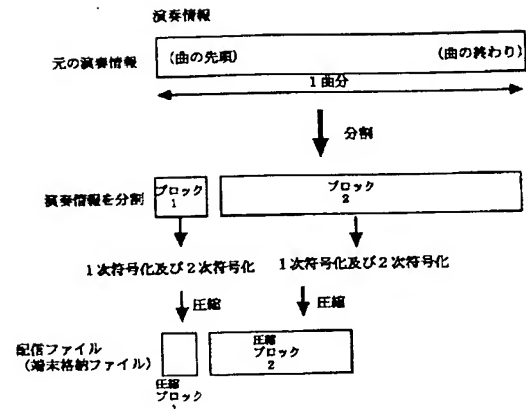
【図 33】



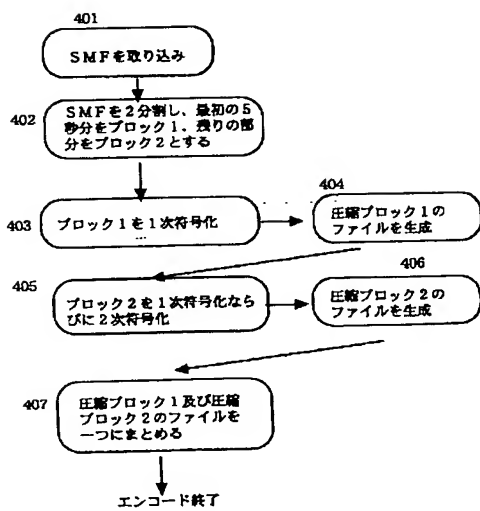
【図27】



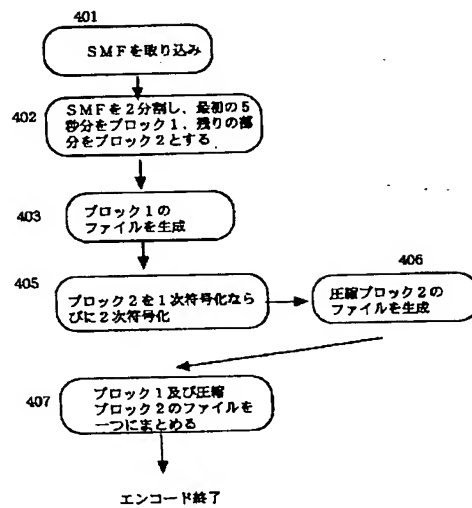
【図34】



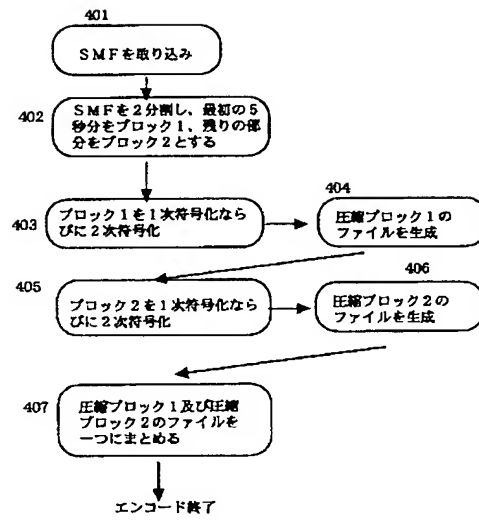
【図35】



【図36】



【図37】



フロントページの続き(51) Int. Cl.⁷

H04M 1/00

識別記号

F I

H04M 1/00

テーマコード(参考)

B